PRINTED COPY IS UNCONTROLLED AND MAY BE OBSOLETE

ipd4600magridrmTES-10

Defense Information Infrastructure (DII) Common Operating Environment (COE)

Application Program Interface Reference Manual (APIRM) for the Grid Field API (MAGRID) Segment of the Tactical Environmental Support System Next Century [TESS(NC)] Meteorology and Oceanography (METOC) Database

Document Version 4.6

29 January 1999

Prepared for:
Naval Research Laboratory
Marine Meteorology Division
Monterey, CA

Prepared by:
Integrated Performance Decisions
Middletown, RI

Table of Contents

1 SCOPE 1
1.1 Identification 1
1.2 System Overview 1

2 REFERENCED DOCUMENTS 5
2.1 Government Documents 5
2.2 Non-Government Documents 6

3 MAGRID OVERVIEW 7
3.1 Overview of the MAGRID Segment 7
3.2 API Overview 7
3.3 Supersession of TEDS APIs 9
3.3.1 MAGRID and TEDS 3.5 API Cross-Reference 9
3.3.2 Obsolete APIs 10
3.4 Grid Field Data Structures/Defines 11
3.4.1 Grid Field Defines/Enumerations 11
3.4.2 Grid Field Area of Interest (AOI) Structure 12
3.4.3 2D Grid Field Ingest Structure 12
3.4.4 3D Grid Field Ingest Structure 13
3.4.5 Grid Field Input Registration Structure 14
3.4.6 Grid Field Projection Description Structures 14
3.4.7 2D Grid Field Query Structure 15
3.4.8 3D Grid Field Query Structure 16
3.4.9 3D Track Query Structure 16
3.4.10 2D Grid Field Catalog Query Return Structure 17
3.4.11 3D Grid Field Catalog Query Return Structure 17
3.4.12 2D Grid Field Retrieve By Reference Structure 18
3.4.13 Grid Field Format Description Structure 18
3.4.14 2D Grid Field Data Structure 18
3.4.15 3D Grid Field Data Structure 19
3.4.16 3D Profile (Stick) Grid Field Data Structure 19
3.4.17 Grid Field Linked List Structure 19
3.4.18 Grid Field Return Structure 19
3.4.19 Registration Output Structure 20
3.4.20 Center Information Data Structure 20
3.4.21 Model Information Data Structure 20
3.4.22 Parameter Information Data Structure 21
3.4.23 Units Information Data Structure 21
3.4.24 Delete Query Data Structure 21
3.5 MAGRID Error Definitions 22

4 CONNECT APIS 25
4.1 MAGRIDConnect 26
4.2 MAGRIDDisconnect 27
4.3 MAGRIDRemoteConnect 28
4.4 MAGRIDRemoteDisconnect 29
4.5 MAGRIDSetConnection 30

5 RETRIEVAL APIS 31
5.1 MAGRIDRetrRegistration 31
5.2 MAGRID2DCatalog 33
5.3 MAGRID3DCatalog 35
5.4 MAGRIDGet2DByID 37
5.5 MAGRIDGet2DByQuery 39
5.6 MAGRIDGetVolumeByID 41
5.7 MAGRIDGetSliceByID 43
5.8 MAGRIDGetProfileByID 45
5.9 MAGRIDGetTrack 46
5.10 MAGRIDGetModelsInfo 47
5.11 MAGRIDGetParametersInfo 49
5.12 MAGRIDGetUnitsInfo 51
5.13 MAGRIDGetCentersInfo 52

6 DATA MANAGEMENT APIS 55
6.1 MAGRID2DIngest 55
6.2 MAGRID3DIngest 57
6.3 MAGRIDDeleteByID 58
6.4 MAGRIDDeleteByQuery 59
6.5 MAGRIDUpdateByID 60
6.6 MAGRIDVerifyModel 61
6.7 MAGRIDRegisterModel 62

7 MAGRID UTILITY METHODS AND FUNCTIONS 63
7.1 MAGRIDFreeLL 63
7.2 MAGRIDGetVolumePtr 64
7.3 MAGRIDGetPoint 65

8 NOTES 67
8.1 Glossary of Acronyms 67

List of Tables

3-1 Grid Field Connect APIs 7
3-2 Grid Field Retrieval APIs 8
3-3 Grid Field Data Management APIs 9
3-4 Grid Field Utility APIs 9
3-5 MAGRID and TEDS API Cross-Reference 10

List of Figures

1-1 TESS(NC) METOC Database Conceptual Organization 3

(This page intentionally left blank.)

1 SCOPE

1.1 Identification

This Application Program Interface (API) Reference Manual (APIRM) describes the APIs provided in the Grid Field API (MAGRID) segment, Version 4.3.0.0, of the Tactical Environmental Support System Next Century [TESS(NC)] Meteorology and Oceanography (METOC) Database. The MAGRID segment provides APIs for the storage and retrieval of Grid Field METOC data. This software is designed to run under the Defense Information Infrastructure (DII) Common Operating Environment (COE), release 3.1, on a Hewlett-Packard computer running HP-UX 10.20 or a personal computer running the Microsoft Windows NT 4.0 operating system with Service Pack 3.

1.2 System Overview

The APIs described in this document form a portion of the METOC Database component of the TESS(NC) Program (Navy Integrated Tactical Environmental Subsystem (NITES) Version I). On 29 October 1996, the Oceanographer of the Navy issued a TESS Program Policy statement in letter 3140 Serial 961/6U570953, modifying the Program by calling for five seamless software versions that are DII COE compliant, preferably to level 5.

The five versions are:

• NITES Version I	The local data fusion center and principal METOC analysis and forecast system (TESS(NC))
• NITES Version II	The subsystem on the Joint Maritime Command Information System (JMCIS) or Global Command and Control System (GCCS) (NITES/Joint METOC Segment (JMS))
• NITES Version III	The unclassified aviation forecast, briefing, and display subsystem tailored to Naval METOC shore activities (currently satisfied by the Meteorological Integrated Data Display System (MIDDS))
• NITES Version IV	The Portable subsystem composed of independent Personal Computers (PCs)/workstations and modules for forecaster, satellite, communications, and Integrated Command, Control, Communications, Computer, and Intelligence Surveillance Reconnaissance (IC4ISR) functions (currently the Interim Mobile Oceanographic Support System (IMOSS))
• NITES Version V	Foreign Military Sales (currently satisfied by the Allied Environmental Support System (AESS))

NITES I acquires and assimilates various METOC data for use by US Navy and Marine Corps weather forecasters and tactical planners. NITES I provides these users with METOC data, products, and applications necessary to support the warfighter in tactical operations and decision making. NITES I provides METOC data and products to NITES I and II applications, as well as non-TESS(NC) systems requiring METOC data, in a heterogeneous, networked computing environment.

The TESS(NC) Concept of Operations and system architecture require that the METOC Database be distributed both in terms of application access to METOC data and products and in terms of physical location of the data repositories. The organizational structure of the database is influenced by these requirements, and the components of this distributed database are described below.

In accordance with DII COE database concepts, the METOC Database is composed of six DII COE-compliant *shared database* segments. Associated with each shared database segment is an API segment. The segments are arranged by data type as follows:

Data Type	**Data Segment**	**API Segment**
Grid Fields	MDGRID	MAGRID
Latitude-Longitude-Time (LLT) Observations	MDLLT	MALLT
Textual Observations and Bulletins	MDTXT	MATXT
Remotely Sensed Data	MDREM	MAREM
Imagery	MDIMG	MAIMG
Climatology Data	MDCLIM	MACLIM

A typical client-server installation is depicted in Figure 1-1 on the next page. This shows the shared database segments residing on a DII COE SHADE database server, with a NITES I or II client machine hosting the API segments. Communication between API segments and shared database segments is accomplished over the network using ANSI-standard Structured Query Language (SQL).

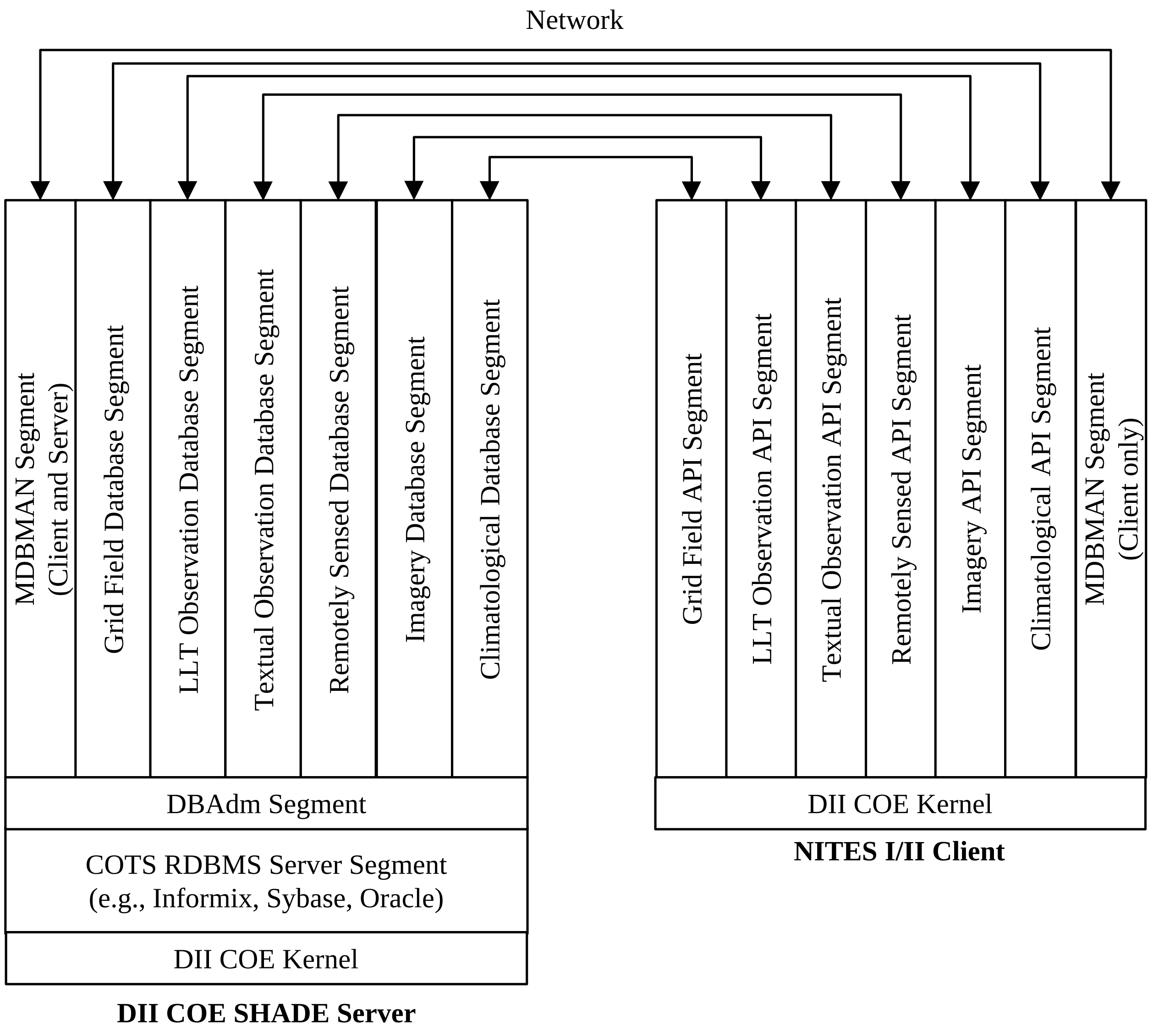
Network
MDBMAN Segment (Client and Server)
Grid Field Database Segment
LLT Observation Database Segment
Textual Observation Database Segment
Remotely Sensed Database Segment
Imagery Database Segment
Climatological Database Segment
DBAdm Segment
COTS RDBMS Server Segment (e.g., Informix, Sybase, Oracle)
DII COE Kernel
DII COE SHADE Server
Grid Field API Segment
LLT Observation API Segment
Textual Observation API Segment
Remotely Sensed API Segment
Imagery API Segment
Climatological API Segment
MDBMAN Segment (Client only)
DII COE Kernel
NITES I/II Client

Figure 1-1. TESS(NC) METOC Database Conceptual Organization

The MAGRID segment deals with gridded METOC datasets. These fields provide forecasters with validation information for various atmospheric and oceanographic parameters. A dataset represents a logical collection of discrete grid field data records. The grid data records are logically organized with each other by center, subcenter, and grid model type. A grid data record contains descriptive information (element, level, forecast period, etc.) and the actual grid values.

(This page intentionally left blank.)

2 REFERENCED DOCUMENTS

2.1 Government Documents

<u>STANDARDS</u>

MIL-STD-498 5 December 1994	*Software Development and Documentation*

<u>SPECIFICATIONS</u>

Unnumbered 27 May 1997	*Performance Specification (PS) for the Tactical Environmental Support System/Next Century TESS(3)/NC (AN/UMK-3)*
Unnumbered 30 September 1997	*Software Requirements Specification for the Tactical Environmental Support System/Next Century [TESS(3)/NC] Meteorological and Oceanographic (METOC) Database*, Space and Naval Warfare Systems Command, Environmental Systems Program Office (SPAWAR PMW-185), Washington, DC

<u>OTHER DOCUMENTS</u>

Unnumbered 02 January 1996	*GRIB (Edition 1)* *The WMO Format for the Storage of Weather Product Information and the Exchange of Weather Product Messages in Gridded Binary Form* U.S. Department of Commerce National Oceanic and Atmospheric Administration National Weather Service National Centers for Environmental Prediction Clifford H. Dey NCEP Central Operations
DII.COE.DocReqs-5 29 April 1997	*Defense Information Infrastructure (DII) Common Operating Environment (COE) Developer Documentation Requirements, Version 1.0*

DII.3010.HP1020.KernelP1.IG-1 9 May 1997	*DII COE Kernel 3.0.1.0P1 Patch 1 for HP-UX 10.20 Installation Guide*
DII.COE31.HP10.20.CIP 23 May 1997	*DII COE V3.1 HP 10.20 Consolidated Installation Procedures*
Unnumbered 20 June 1997	*Tactical Environmental Data System (TEDS) Release 3.5 Model Data Application Program Interface (API) User's Guide*
DII.3010.HP1020.KernelP2.IG-1 30 July 1997	*DII COE Kernel 3.0.1.0P2 Patch 2 for HP-UX 10.20 Installation Guide*
DII.3010.HP1020.KernelP3.IG-1 08 August 1997	*DII COE Kernel 3.0.1.0P3 Patch 3 for HP-UX 10.20 Installation Guide*
DII.3010.HP1020.KernelP4.IG-1 27 August 1997	*DII COE Kernel 3.0.1.0P4 Patch 4 for HP-UX 10.20 Installation Guide*
Unnumbered 30 September 1997	*Database Design Description for the Tactical Environmental Support System/Next Century [TESS(3)/NC)] Meteorological and Oceanographic (METOC) Database,* Space and Naval Warfare Systems Command, Environmental Systems Program Office (SPAWAR PMW-185), Washington, DC
ipd4600magridpmTES-10 29 January 1999	*Programming Manual (PM) for the Grid Field API (MAGRID) Segment of the Tactical Environmental Support System Next Century [TESS(NC)] Meteorology and Oceanography (METOC) Database*

2.2 Non-Government Documents

World Meteorological Organization, Geneva, Switzerland

WMO-306	*Manual On Codes*

3 MAGRID OVERVIEW

3.1 Overview of the MAGRID Segment

The MAGRID segment was designed to manage all aspects of both 2D and 3D grids. As such, there are some APIs that support only 2D grid methods, some that support 3D grid methods, and others that support the management of common elements within the database (such as model registration).

3.2 API Overview

The APIs are organized into four categories: Connect, Retrieval, Data Management, and Utility. The routines that deal with connecting, disconnecting, and changing connections are grouped under Connect. The Retrieval routines are used to retrieve grids, registrations, or table information (e.g., Model table). The Data Management routines are used to ingest, delete, and update grids in the database. Finally, several utility routines are used to manipulate the data. Table 3-1 through Table 3-4 provide a list of all APIs for Grid Field data.

Table 3-1. Grid Field Connect APIs

API Name	Functional Description
MAGRIDConnect	Connects the application to the database.
MAGRIDDisconnect	Disconnects the application from the database.
MAGRIDRemoteConnect	Allows the application to connect to a database residing on a remote server and/or to establish multiple connections.
MAGRIDRemoteDisconnect	Disconnects the application from the database residing on a remote server.
MAGRIDSetConnection	Sets the connection context for multiple connections to different database servers and/or databases.

Table 3-2. Grid Field Retrieval APIs

API Name	Functional Description
MAGRID2DCatalog	Retrieves a catalog of 2D grids.
MAGRID3DCatalog	Retrieves a catalog of 3D grids.
MAGRIDGet2DByID	Retrieves a single 2D grid from the database given the datasetname, recordId, and desired format of the data.
MAGRIDGet2DByQuery	Retrieves one or more 2D grids from the database that match the specified query criteria.
MAGRIDGetCentersInfo	Retrieves a linked list of selected center descriptive information.
MAGRIDGetModelsInfo	Retrieves a linked list of selected model descriptive information.
MAGRIDGetParametersInfo	Retrieves a linked list of selected parameter descriptive information.
MAGRIDGetProfileByID	Retrieves a single 3D grid profile from the database given the dataset name, record id, latitude, and longitude position.
MAGRIDGetSliceByID	Retrieves a single 3D grid horizontal slice from the database given the dataset name, record id, level, and desired format of the data.
MAGRIDGetTrack	Retrieves a track of 3D grid profiles from the database given the starting latitude/longitude point, range, bearing, and resolution.
MAGRIDGetUnitsInfo	Retrieves a linked list of selected unit descriptive information.
MAGRIDGetVolumeByID	Retrieves a single 3D grid from the database given the dataset name, record id, and desired format of the data.
MAGRIDRetrRegistration	Retrieves a linked list of registration information.

Table 3-3. Grid Field Data Management APIs

API Name	Functional Description
MAGRID2DIngest	Adds a 2D Grid Field record to the database.
MAGRID3DIngest	Adds a 3D Grid Field record to the database.
MAGRIDDeleteByID	Deletes a record from a dataset (2D and 3D) in the database.
MAGRIDDeleteByQuery	Deletes multiple grid records from a specified dataset based on query criteria supplied.
MAGRIDRegisterModel	Registers a grid model into the database.
MAGRIDUpdateByID	Updates grid field points in a record from a dataset (2D and 3D) in the database.
MAGRIDVerifyModel	Determines if the registration information exists for a given grid, center subcenter, and generating process.

Table 3-4. Grid Field Utility APIs

API Name	Functional Description
MAGRIDFreeLL	Frees the linked list associated with catalog, grid, and registration retrieval.
MAGRIDGetPoint	Retrieve a single point from a given 2D grid field and registration.
MAGRIDGetVolumePtr	Get a pointer into the 3D BLOB Data for a given level.

3.3 Supersession of TEDS APIs

3.3.1 MAGRID and TEDS 3.5 API Cross-Reference

The MAGRID APIs replace the older Tactical Environmental Data System (TEDS) APIs, documented in the *TEDS Release 3.5 Model Data API User's Guide* referenced in Section 2 of this document. Applications using the TEDS APIs will need to be rewritten to use the equivalent MAGRID APIs. Table 3-5 provides a cross-reference of the MAGRID and TEDS APIs for the convenience of developers.

The following important considerations should be noted:

- In some cases, a single MAGRID API replaces several TEDS APIs or vice versa.
- Because the data structures differ between the two implementations, it is imperative that care be taken to ensure correct conversion.

Table 3-5. MAGRID and TEDS API Cross-Reference

TEDS API(s)	MAGRID API
ted_start	MAGRIDConnect
ted_stop	MAGRIDDisconnect
ted_GridRetrRef	MAGRIDGet2DByID
ted_GridRetr	MAGRIDGet2DByQuery
ted_GridStore	MAGRID2DIngest
ted_RetrCat	MAGRID2DCatalog
ted_Retrieve3D	MAGRID3DGetVolumeByID
ted_3DRetr	MAGRID3DCatalog, MAGRID3DGetVolumeByID
ted_Store3D	MAGRID3DIngest

3.3.2 Obsolete APIs

The following list of APIs will not be supported in the next delivery of the MAGRID segment. References to these APIs have been removed from the APIRM. The libraries will continue to support these APIs until the next delivery.

Obsolete		Superseded By
MAGRIDStore	–>	MAGRID2DIngest
MAGRIDRegister2DModel	–>	MAGRIDRegisterModel
MAGRIDVerify2DRegistration	–>	MAGRIDVerifyModel
MAGRIDRetrCat	–>	MAGRID2DCatalog
MAGRIDRetrRef	–>	MAGRIDGet2DByID
MAGRIDRetr	–>	MAGRIDGet2DByQuery
MAGRIDDelete	–>	MAGRIDDeleteByID

3.4 Grid Field Data Structures/Defines

The following structures are used by the Grid Field APIs and are provided for reference:

3.4.1 Grid Field Defines/Enumerations

```
#define MAGRID_DATASET_LEN            18      /* Max Length of Dataset Name      */
#define MAGRID_NO_UNIT_CONVERSION     -1      /* Denotes no unit conversion      */
#define MAGRID_QUERY_WILDCARD         -9999   /* wildcard value for query fields */
#define MAGRID_QUERY_WILDCARD_STRING ”-9999” /* wildcard string                 */
#define MAGRID_COMMON_PARAMETER       0       /* Parameters common to all Centers*/
#define MAGRID_UNDEFINED_GRIDID       255     /* Denotes undefined GRIB Id       */
#define MAGRID_MAX_3D_LEVEL           75      /* Max number of 3D Levels         */
#define MAGRID_REGISTRATION_NAME_LEN  32      /* Max Length of Registration Name */
#define MAGRID_MODEL_NAME_LEN         64      /* Max Length of Model Name        */
#define MAGRID_CENTER_NAME_LEN        64      /* Max Length of Center Name       */
#define MAGRID_PARM_NAMLEN            64      /* Max Length of Parameter Name    */
#define MAGRID_UNIT_NAMLEN            64      /* Max Length of Unit Name         */
#define MAGRID_UNITABR_NAMLEN         32      /* Max Length of Unit Abbreviation */
#define MAGRID_MISSING_VALUE          -9999.0 /* Missing Grid Point Value        */
#define MAGRID_DEFAULT_CONNECTION     “mdgrid_default_connection”
#define MAGRID_LEVEL_LEN              12      /* Max Length of Level Name        */
#define MAGRID_CLASS_LEN              32      /* Max Length of Classification    */
#define MAGRID_RMETHOD_LEN            32      /* Max Length of Receipt Method    */
#define MAGRID_COMPRESS_LEN           32      /* Max Length of Compression       */
#define MAGRID_DESCRIPTION_LEN        32      /* Max Length of Description       */
#define MAGRID_EOF_COMPRESSION        “EOF”   /* Setting for szCompression field */

/***************************************************************/
/* Output Format Values                                        */
/***************************************************************/
typedef enum {
   MAGRID_GET_AS_SPECIFIED = 0
   MAGRID_GET_AS_STORED    = 1
} EMAGRIDOUTPUTFORMATS;

/***************************************************************/
/* Scan Mode Values                                            */
/***************************************************************/
typedef enum {
   MAGRID_pXinnY=1,
   MAGRID_pXinpY,
   MAGRID_nYinpX,
   MAGRID_pYinpX
} EMAGRIDSTORDSC;

/***************************************************************/
/* Projection Values                                           */
/***************************************************************/
typedef enum {
   MAGRID_NO_PROJECTION    = (-1),
   MAGRID_POLAR            = 1,
   MAGRID_LAMBERT          = 2,
   MAGRID_MERCATOR         = 4,
   MAGRID_SPHERICAL_PROJ   = 16
} EMAGRIDPROJECTIONS;
```

```
/***************************************************************/
/* DataCategory Values                                         */
/***************************************************************/
typedef enum {
   MAGRID_GET_ALL = MAGRID_QUERY_WILDCARD
   MAGRID_BASE    = 0,
   magrid_edited  = 1,
   MAGRID_DERIVED = 2
} EMAGRIDCATEGORIES;

/***************************************************************/
/* GridType Values                                             */
/***************************************************************/
typedef enum {
   MAGRID_BOTH  = 0,
   MAGRID_2D    = 1,
   MAGRID_3D    = 2,
} EMAGRIDTYPE;
```

3.4.2 Grid Field Area of Interest (AOI) Structure

This structure defines the geographical AOI. This structure is defined in `MAGRIDAPI.h`.

```
typedef struct tagmagridGeoArea {
   float rsNLat;                          /* North Latitude */
   float rsSLat;                          /* South Latitude */
   float rsWLon;                          /* West Longitude */
   float rsELon;                          /* East Longitude */
} MAGRIDGEOAREA, *PMAGRIDGEOAREA;
```

3.4.3 2D Grid Field Ingest Structure

This structure defines the data required during ingest. This structure is defined in `MAGRIDAPI.h.`

```
typedef struct tagmagrid2DIngest
{
  long      lGeneratingProcId;              /* Generating Process (Model) ID*/
                                            /* Range 1-255                  */
  long      lGridId;                        /* Registration ID              */
                                            /* Range 1-9999                 */
  long      lProductionCenterId;            /* Production center ID         */
                                            /* Range 1-99                   */
  long      lSubCenterId;                   /* Sub center ID                */
                                            /* Range 0-255                  */
  long      lParameterId;                   /* Parameter ID                 */
                                            /* Range 1-255                  */
  long      lParameterUnit;                 /* Parameter unit               */
                                            /* Range 0-2**31                */
  long      lLevelType;                     /* Level Type                   */
                                            /* Range 1-255                  */
  long      lBaseTime;                      /* Model Basetime in epoch time */
                                            /* Range 0-2**31                */
  long      lTau;                           /* Forecast hour (minutes)      */
                                            /* Range 0-14400 minutes        */
  float     rsLevelLo;                          /* Lowest Level is represented  */
  float     rsLevelHi;                          /* Highest Level is represented */
  long      lQualityIndicator;                  /* pass/failure quality checks  */
  EMAGRIDCATEGORIES eDataCategory;              /* 0=Base, 1=Edited, 2=Derived  */
  char      szSecurityClass[32];                /* Security Classification      */
  char      szReceiptMethod[32];                /* Communications Interface     */
  char      szCompression[32];                  /* Compression Indicator        */
  char      szDescription[MAGRID_DESCRIPTION_LEN]; /* Description               */
  unsigned  long ulSize;                        /* Size of grid data (bytes)    */
  float     *pGridFieldData;                    /* Ptr to the grid field data   */
  char      szDataSetName[MAGRID_DATASET_LEN+1];   /* DataSetName               */
  long      lRecordId;                          /* Record Id within Dataset     */
} MAGRID2DINGEST, *PMAGRID2DINGEST, MAGRIDSTORE, *PMAGRIDSTORE;
```

3.4.4 3D Grid Field Ingest Structure

This structure defines the data required during 3D Grid Field ingest. This structure is defined in `MAGRIDAPI.h`.

```
typedef struct tagmagrid3DIngest
{
  long    lGeneratingProcId;                    /* Model name and ID            */
                                                /* Range 1-255                  */
  long    lGridId;                              /* registration name and ID     */
                                                /* Range 1-9999                 */
  long    lProductionCenterId;                  /* production center name and ID */
                                                /* Range 1-99                   */
  long    lSubCenterId;                         /* sub center name and ID       */
                                                /* Range 0-255                  */
  long    lParameterId;                         /* parameter name and ID        */
                                                /* Range 1-255                  */
  long    lParameterUnit;                       /* parameter unit type          */
                                                /* Range 0-2**31                */
  float   rsLevel[MAGRID_MAX_3D_LEVEL];         /* Levels                       */
  long    lLevelType;                           /* Level Type                   */
                                                /* Range 1-255
  long    lMaxZPoint;                           /* Number of Levels             */
                                                /* Range 2-75                   */
  long    lBaseTime;                            /* Model Basetime and epoch time */
                                                /* Range 0-2**31
  long    lTau;                                 /* Forecast hour (minutes)      */
                                                /* Range 0-14400 minutes        */
  long    lQualityIndicator;                    /* pass/failure quality checks  */
  EMAGRIDCATEGORIES eDataCategory;              /* 0=Base, 1=Edited, 2=Derived  */
  char    szSecurityClass[32];                  /* Security Classification      */
  char    szReceiptMethod[32];                  /* Communications Interface     */
  char    szCompression[32];                    /* Compression Indicator        */
  char    szDescription[MAGRID_DESCRIPTION_LEN];/* Description                  */
  unsigned long ulSize;                         /* Size of grid data (bytes)    */
  float   *pGridFieldData;                      /* Ptr to the grid field data   */
  char    szDataSetName[MAGRID_DATASET_LEN+1];  /* DataSetName                  */
  long    lRecordId;                            /* Record Id within Dataset     */
} MAGRID3DINGEST,*PMAGRID3DINGEST;
```

3.4.5 Grid Field Input Registration Structure

This structure defines the data required to register a model geometry with the database. This structure is defined in `MAGRIDAPI.h`.

```
typedef struct tagmagridInputReg
{
  long      lGridId;             /* Registration Id                          */
                                 /* Range 1-9999                             */
  long      lProductionCenterId; /* Production Center ID                     */
                                 /* Range 1-99                               */
  long      lSubCenterId;        /* sub center name and ID                   */
                                 /* Range 0-255                              */
  long      lGeneratingProcId;   /* Model name and ID                        */
                                 /* Range 1-255                              */
  float     rsRegLat;            /* Registered lat corresponding to rsRegY  */
                                 /* Range -90 through 90                     */
  float     rsRegLon;            /* Registered long corresponding to rsRegX */
                                 /* Range -180 through 360                   */
  float     rsRegX;              /* X coordinate corresponding to rsRegLon  */
  float     rsRegY;              /* Y coordinate corresponding to rsRegLat  */
  float     rsXDistance;         /* X Grid spacing in km                     */
  float     rsYDistance;         /* Y Grid spacing in km                     */
  long      lMaxXPoint;          /* Number of X Grid Points                  */
                                 /* Range 2-10000                            */
  long      lMaxYPoint;          /* Number of Y Grid Points                  */
                                 /* Range 2-10000                            */
  long      lMaxZPoint;          /* Number of Z Grid Points                  */
                                 /* Range 2-75                               */
  EMAGRIDSTORDSC   eScanMode;    /* ScanMode                                 */
  MAGRIDPROJDESC  stProjectionDesc;
  char      szRegName [MAGRID_REGISTRATION_NAME_LEN];
  char      szModelName[MAGRID_MODEL_NAME_LEN];
  char      szCenterName[MAGRID_CENTER_NAME_LEN];
} MAGRIDINPUTREG, *PMAGRIDINPUTREG;
```

3.4.6 Grid Field Projection Description Structures

```
/**************************************************************/
/*Projection Indicator and Union Structure for supported Projections.*/
/**************************************************************/
typedef struct tagmagridProjectionDesc {
   EMAGRIDPROJECTIONS eProjection;
   union  { MAGRIDLAMBERTMERC lamberConf;
            MAGRIDLAMBERTMERC mercator;
            MAGRIDSPHERICAL   spherical;
            MAGRIDPOLARSTEREO polarStereo;
          } projParms;
}MAGRIDPROJDESC, *PMAGRIDPROJDESC;

/**************************************************************/
/* Lambert Conformal and Mercator registration parameters structure. */
/**************************************************************/
typedef struct tagmagridLambertMerc /*Fields needed to describe a Lambert
                                      Conformal or Mercator grid           */
{
   float rsStandardLat1;            /*Northern most latitude of projection  */
   float rsStandardLat2;           /*Southern most latitude of projection  */
   float rsStandardLon;            /*Longitude which bisects Northern and
                                     Southern Parallels.                   */
} MAGRIDLAMBERTMERC, *PMAGRIDLAMBERTMERC;
/*************************************************************
  Spherical registration parameters structure.
*************************************************************/
typedef struct tagmagridSpherical /*Fields needed to describe spherical grid*/
{
   float rsXResolution;            /* Distance, in degrees decimal, between
                                     longitudinal grid points             */
   float rsYResolution;            /* Distance, in degrees decimal, between
                                     latitudinal grid points              */
} MAGRIDSPHERICAL, *PMAGRIDSPHERICAL;
/*************************************************************
  Polar Stereographic registration parameters structure.
*************************************************************/
typedef struct tagmagridPolarStereo /* Fields needed to describe
                                     a polar grid                          */
{
   float rsStandardLat;            /*Latitude to which projection extends  */
   float rsStandardLon;            /*Longitude to which projection is
                                     oriented                              */
} MAGRIDPOLARSTEREO, *PMAGRIDPOLARSTEREO;
```

3.4.7 2D Grid Field Query Structure

The following is an input structure used to submit a 2D catalog/grid query. All fields must be either set or wildcarded. The AOI cannot be wildcarded. If lParameterId is wildcarded, then lProductionCenterId and lSubCenterId cannot be wildcarded. This structure is defined in the file `MAGRIDAPI.h.`

```
typedef struct tagmagridQuery
{
   long  lGeneratingProcId;        /* WMO GRIB Generating Process (Model) Id  */
   long  lProductionCenterId;      /* WMO GRIB Center Id                      */
   long  lSubCenterId;             /* WMO GRIB SubCenter Id                   */
   long  lGridId;                  /* GridID (Registration)                   */
   long  lParameterId;             /* WMO GRIB Parameter Id                   */
   long  lBeginBaseTime;           /* Beginning Model Basetime in epoch time  */
   long  lEndBaseTime;             /* Ending Model Basetime in  epoch time    */
   long  lBeginTau;                /* Begin Forecast Period Range (0 - 14400
                                      minutes)                                */
   long  lEndTau;                  /* Ending Forecast Period Range (0 – 14400
                                      minutes)                                */
   float rsBeginLevel;             /* Begin levels to be queried              */
   float rsEndLevel;               /* End levels to be queried                */
   long  lLevelType;               /* WMO GRIB Level Type                     */
   MAGRIDGEOAREA  stGeoArea;       /* Aoi Description                         */
   long  lQualityIndicator;        /* pass/failure quality checks             */
   EMAGRIDCATEGORIES eDataCategory; /* 0=Base, 1=Edited, 2=Derived            */
   char  szCompression[32];        /* Compression Indicator                   */
   char  szDescription[32];        /* Description Field                       */
   long  lBeginReceiptTime;        /* Begin Receipt Time                      */
   long  lEndReceiptTime;          /* End Receipt Time                        */
} MAGRIDQUERY, *PMAGRIDQUERY;
```

3.4.8 3D Grid Field Query Structure

The following is an input structure used to submit a 3D catalog query. All fields must be either set or wildcarded. The AOI cannot be wildcarded. If lParameterId is wildcarded, then lProductionCenterId and lSubCenterId cannot be wildcarded. This structure is defined in the file `MAGRIDAPI.h.`

```
typedef struct tagmagrid3DQuery
{
   long  lGeneratingProcId;         /* WMO GRIB Generating Process (Model) Id */
   long  lProductionCenterId;       /* WMO GRIB Center Id                     */
   long  lSubCenterId;              /* WMO GRIB SubCenter Id                  */
   long  lGridId;                   /* GridId (Registration)                  */
   long  lParameterId;              /* WMO GRIB Parameter Id                  */
   long  lBeginBaseTime;            /* Beginning Model Basetime in epoch time */
   long  lEndBaseTime;              /* Ending Model Basetime in  epoch time   */
   long  lBeginTau;                 /* Begin Forecast Period Range (0-14400
                                       minutes)                               */
   long  lEndTau;                   /* Ending Forecast Period Range (0–14400
                                       minutes)                               */
   MAGRIDGEOAREA  stGeoArea;        /* Aoi Description                        */
   long  lQualityIndicator;         /* pass/failure quality checks            */
   EMAGRIDCATEGORIES eDataCategory; /* 0=Base, 1=Edited, 2=Derived            */
   char  szCompression[32];         /* Compression Indicator                  */
   char  szDescription[32];         /* Description Field                      */
   long  lBeginReceiptTime;         /* Begin Receipt Time                     */
   long  lEndReceiptTime;           /* End Receipt Time                       */
} MAGRID3DQUERY, *PMAGRID3DQUERY;
```

3.4.9 3D Track Query Structure

The following is an input structure used to submit a 3D track query. All fields must be either set or wildcarded. The AOI cannot be wildcarded. If lParameterId is wildcarded, then lProductionCenterId and lSubCenterId cannot be wildcarded. This structure is defined in the file `MAGRIDAPI.h.`

```
typedef struct tagmagrid3DTrackQuery
{
   long  lGeneratingProcId;   /*WMO GRIB Generating Process (Model) Id      */
   long  lProductionCenterId; /* WMO GRIB Center Id                         */
   long  lSubCenterId;        /* WMO GRIB SubCenter Id                      */
   long  lGridId;             /* GridId (Registration) Id                   */
   long  lParameterId;        /* WMO GRIB Parameter Id                      */
   long  lBeginBaseTime;      /*Beginning Model Basetime in epoch time      */
   long  lEndBaseTime;        /* Ending Model Basetime in  epoch time       */
   float rsRange;             /* range of track in nautical miles           */
   float rsBearing;           /* bearing of track in degrees                */
   float rsLat;               /* starting latitude                          */
   float rsLon;               /* starting longitude                         */
   long  nResolution;         /* track resolution (number of desired        */
                              /* profiles.                                  */
   MAGRIDGEOAREA  stGeoArea;  /* Aoi Description                            */
} MAGRID3DTRACKQUERY, *PMAGRID3DTRACKQUERY;
```

3.4.10 2D Grid Field Catalog Query Return Structure

The following structure is returned from a 2D catalog query. This structure is defined in `MAGRIDAPI.h`.

```
typedef struct tagmagridFieldDesc
{
   char szDataSetName[MAGRID_DATASET_LEN+1];   /* DataSetName                  */
   long lRecordId;                             /* Record Id within Dataset     */
   long lGridId,                               /* WMO GRIB Grid Id             */
        lBaseTime,                             /* Model Basetime  (epoch)      */
        lGeneratingProcId,                     /* Generating Process Id (model) */
        lProductionCenterId,                   /* WMO GRIB center  Id          */
        lSubCenterId,                          /* WMO GRIB Sub center Id       */
        lParameterId,                          /* WMO GRIB Parameter  Id       */
        lParameterUnit,                        /* Parameter unit  Id           */
        lLevelType;                            /* WMO GRIB Level Type Id       */
  long  llReceiptTime;                         /* Dataset creation time        */
  long  lTau;                                  /* Forecast hour                */
  long  lMaxXPoint;                            /* Max X grid dimension         */
  long  lMaxYPoint;                            /* Max Y grid dimension         */
  float rsLevelLo;                             /* Lowest  Level is represented */
  float rsLevelHi;                             /* Highest Level is represented */
  long  lQualityIndicator;                     /* pass/failure quality checks  */
 EMAGRIDCATEGORIES eDataCategory;              /* 0=Base, 1=Edited, 2=Derived  */
 EMAGRIDPROJECTIONS eProjection;               /* Projection                   */
 EMAGRIDSTORDSC eScanMode;                     /* Scan Mode                    */
 MAGRIDGEOAREA stGeoArea;                      /* Aoi Description              */
  char  szSecurityClass[32];                   /* Security Classification      */
  char  szReceiptMethod[32];                   /* Communications Interface     */
  char  szCompression[32];                     /* Compression Indicator        */
  char  szDescription[MAGRID_DESCRIPTION_LEN]; /* Description                  */
} MAGRIDFIELDDESC, *PMAGRIDFIELDDESC;
```

3.4.11 3D Grid Field Catalog Query Return Structure

The following structure is returned from a 3D catalog query. This structure is defined in `MAGRIDAPI.h`.

```
typedef struct tagmagrid3DFieldDesc
{
   char szDataSetName[MAGRID_DATASET_LEN+1];   /* DataSetName                  */
   long lRecordId;                             /* Record Id within Dataset     */
   long lGridId,                               /* WMO GRIB Grid Id             */
        lBaseTime,                             /* Model Basetime  (epoch)      */
        lGeneratingProcId,                     /* Generating Process Id (model) */
        lProductionCenterId,                   /* WMO GRIB center  Id          */
        lSubCenterId,                          /* WMO GRIB Sub center Id       */
        lParameterId,                          /* WMO GRIB Parameter  Id       */
        lParameterUnit,                        /* Parameter unit  Id           */
  long  lReceiptTime;                          /* Dataset creation time        */
  long  lTau;                                  /* Forecast hour                */
  long  lMaxXPoint;                            /* Max X grid dimension         */
  long  lMaxYPoint;                            /* Max Y grid dimension         */
  long  lMaxZpoint;                            /* Max level dimension-number of
                                                  levels                       */
  float rsLevel[MAX_3D_LEVEL];                 /* Levels                        */
  long  lLevelType;                            /* WMO GRIB Level Type Id        */
  long  lQualityIndicator;                     /* pass/failure quality checks   */
  EMAGRIDCATEGORIES eDataCategory;             /* 0=Base, 1=Edited, 2=Derived   */
  EMAGRIDPROJECTIONS eProjection;              /* Projection                    */
  EMAGRIDSTORDSC eScanMode;                    /* Scan Mode                     */
  MAGRIDGEOAREA stGeoArea;                     /* Aoi Description               */
  char  szSecurityClass[32];                   /* Security Classification       */
  char  szReceiptMethod[32];                   /* Communications Interface      */
  char  szCompression[32];                     /* Compression Indicator         */
  char  szDescription[MAGRID_DESCRIPTION_LEN]; /* Description                   */
} MAGRID3DFIELDDESC, *PMAGRID3DFIELDDESC;
```

3.4.12 2D Grid Field Retrieve By Reference Structure

The following input structure is used to retrieve a 2D or 3D grid from the database. This structure is defined in `MAGRIDAPI.h`.

```
typedef struct tagmagridReference
{
   char szDataSetName[MAGRID_DATASET_LEN+1]; /* DataSetName                 */
   long lRecordId;                           /* Record Id within Dataset    */
   MAGRIDGEOAREA   stGeoArea;                /* Aoi Description             */
   MAGRIDFORMAT    stGridFormat;             /* User Requests for return data*/
} MAGRIDREFERENCE, *PMAGRIDREFERENCE;
```

3.4.13 Grid Field Format Description Structure

The following is an input structure used to specify the format of the retrieved 2D or 3D grid. This structure is defined in `MAGRIDAPI.h.`

```
typedef struct tagmagridFormat
{
   EMAGRIDOUTPUTFORMATS  eOutputFormat
   long                  lMaxXPoint;
   long                  lMaxYPoint;
   EMAGRIDSTORDSC        eScanMode;
   MAGRIDPROJDESC        stProjectionDesc;
   long                  lUnitId;          /* Unit in which to return data
                                              or NO_UNIT_CONVERSION        */
} MAGRIDFORMAT, *PMAGRIDFORMAT;
```

3.4.14 2D Grid Field Data Structure

The following structure contains the retrieved 2D grid data. This structure is defined in `MAGRIDAPI.h`.

```
typedef struct tagmagridData
{
   MAGRIDFIELDDESC stGridFieldDesc;  /* Description of Grid Retrieved       */
   unsigned long ulSize;             /* Size of grid data buffer (bytes)    */
   float *pGridFieldData;            /* Pointer to the grid field data      */
} MAGRIDDATA, *PMAGRIDDATA;
```

3.4.15 3D Grid Field Data Structure

The following structure contains the retrieved 3D grid volume data. This structure is defined in `MAGRIDAPI.h`.

```
typedef struct tagmagrid3DData
{
   MAGRID3DFIELDDESC stGridFieldDesc;  /* Description of Grid Retrieved     */
   unsigned long ulSize;               /* Size of grid data buffer (bytes)  */
   float *pGridFieldData;              /* Pointer to the grid field data    */
} MAGRID3DDATA, *PMAGRID3DDATA;
```

3.4.16 3D Profile (Stick) Grid Field Data Structure

The following structure contains the retrieved 3D grid profile data. This structure is defined in `MAGRIDAPI.h`.

```
typedef struct tagmagrid3DStick
{
   MAGRID3DFIELDDESC stGridFieldDesc;  /* Description of Grid Retrieved     */
   float rsLat;                        /* Latitude position of profile      */
   float rsLon;                        /* Longitude position of profile     */
   unsigned long ulSize;               /* Size of grid data buffer (bytes)  */
   float *pGridFieldData;              /* Pointer to the grid field data    */
} MAGRID3DSTICK, *PMAGRID3DSTICK;
```

3.4.17 Grid Field Linked List Structure

This is a generic structure used to keep linked lists. This structure is defined in `MAGRIDAPI.h`.

```
typedef struct tagmagridLinkedList {
        void *next;                 /* Pointer to next Link                 */
        void *prev;                 /* Pointer to the parent Link           */
        void *data;                 /* Pointer to the actual data           */
        int     nInternalMask;      /* Internal usage only                  */
} MAGRIDLINKEDLIST, *PMAGRIDLINKEDLIST;
```

3.4.18 Grid Field Return Structure

Each of the Grid Field APIs returns this structure containing status data. The *nStatus* field will contain a zero upon successful completion of the API call. If the field is set to one, there is an SQL error, and the szSQLState field must be examined. Any value of nStatus greater than one indicates a MAGRID error that maps to a define in the file `MAGRIDErr.h`. The Grid Field return structure is defined in `MAGRIDRet.h`.

```
typedef struct tagMAGRIDret
{
   int    nStatus;       /* if nStatus == 0 ==> Success                      */
```

```
                        /* if nStatus == 1 ==> Check SQLState            */
                        /* if nStatus >= 2 ==> Segment specific error      */
   char   szSQLState[6]; /* First  2 characters represent the class,
                            last 3 characters the subclass               */
                        /* where an SQL error occurred.                  */
   char   szErrorMessage[290];
                        /* String containing information about the error   */
}  MAGRIDRET, *PMAGRIDRET;
```

3.4.19 Registration Output Structure

The following structure contains definitions of all the fields of the MAGRID Registrations table. It is returned as part of the registration retrieval processing.

```
typedef struct tagmagridOutputReg {
   float         rsRegLat;     /* Registered latitude corresponding to rsRegY*/
   float         rsRegLon;     /* Registered longitude corresponding to
                                  rsRegX                                   */
   float         rsRegX;       /* X coordinate corresponding to rsRegLon   */
   float         rsRegY;       /* Y coordinate corresponding to rsRegLat   */
   float         rsXDistance;  /* X Grid spacing in km
   float         rsYDistance;  /* Y Grid spacing in km
   long          lMaxXPoint;   /* Maximum X Point (numbered from 1 - n )   */
   long          lMaxYPoint;   /* Maximum Y Point (numbered from 1 - n )   */
   long          lMaxZPoint;   /* Maximum Z Point (numbered from 1 - n )   */
   MAGRIDPROJDESC stProjectionDesc;
   EMAGRIDSTORDSC  eScanMode;
   long          lAoiID;       /* AOI Id                                   */
   long          lGridId;      /* Grid Id                                  */
   long          lCenterId;    /* Production Center Id                     */
   long          lSubCenterId; /* Sub Center Id                            */
   char          szRegName[MAGRID_REGISTRATION_NAME_LEN];
   MAGRIDGEOAREA stGeoArea;    /* AOI description                          */
}  MAGRIDOUTPUTREG, *PMAGRIDOUTPUTREG;
```

3.4.20 Center Information Data Structure

The following structure is returned via a MAGRIDLINKEDLIST structure by the MAGRIDGetCentersInfo API call.

```
typedef struct tagMAGRIDCenterData
{
   long        lProductionCenterId;  /* production center name and ID       */
   long        lSubCenterId;         /* sub center name and ID              */
   char szCenterName[MAGRID_CENTER_NAME_LEN];
} MAGRIDCENTERDATA, *PMAGRIDCENTERDATA;
```

3.4.21 Model Information Data Structure

The following structure is returned via a MAGRIDLINKEDLIST structure by the MAGRIDGetModelsInfo API call.

```
typedef struct tagMAGRIDModelData
```

```
{
   long        lProductionCenterId;  /* production center name and ID       */
   long        lSubCenterId;         /* sub center name and ID              */
   long        lGeneratingProcId;    /* Model ID                            */
   long        lGridId;              /* registration ID                     */
   char        szCenterName[MAGRID_CENTER_NAME_LEN];
   char        szModelName[MAGRID_MODEL_NAME_LEN];
} MAGRIDMODELDATA, *PMAGRIDMODELDATA;
```

3.4.22 Parameter Information Data Structure

The following structure is returned via a MAGRIDLINKEDLIST structure by the MAGRIDGetParametersInfo API.

```
typedef struct tagMAGRIDParameterData
{
   long        lProductionCenterId;  /* production center name and ID       */
   long        lSubCenterId;         /* sub center ID                       */
   long        lParameterId;         /* parameter ID                        */
   char        szParameterName[MAGRID_PARM_NAMLEN];
   float       rsMinValidValue;      /* Minimum Valid Unit value            */
   float       rsMaxValidValue;      /* Maximum Valid Unit value            */
   long        lDefaultUnitId;       /* Unit Type identifier                */
   char        szDefaultUnitAbrv[MAGRID_UNITABR_NAMLEN];
   char        szDefaultUnitName[MAGRID_UNIT_NAMLEN];
} MAGRIDPARAMETERDATA, *PMAGRIDPARAMETERDATA;
```

3.4.23 Units Information Data Structure

The following structure is returned via a MAGRIDLINKEDLIST structure by the MAGRIDGetUnitsInfo API.

```
typedef struct tagMAGRIDUnitData
{
   long        lUnitId;
   char        szUnitAbrv[MAGRID_UNITABR_NAMLEN];
   char        szUnitName[MAGRID_UNIT_NAMLEN];
} MAGRIDUNITDATA, *PMAGRIDUNITDATA;
```

3.4.24 Delete Query Data Structure

The following is an input structure used to submit a delete grid query. The lGridId, lGeneratingProcId, lProductionCenterId, and lSubCenterId may not be wildcarded. The eGridType field may only be MAGRID_2D or MAGRID_3D. The rest of the fields may be wildcarded.

```
typedef struct tagmagridBrowseQuery
{
   EMAGRIDTYPE eGridType; /* MAGRID_2D or MAGRID_3D */
   long      lGridId;                /* Grid Identifier                     */
   long      lGeneratingProcId;      /* Model name and ID                   */
   long      lProductionCenterId;     /* production center name and ID      */
   long      lSubCenterId;            /* sub center name and ID             */
   long      lParameterId;            /* parameter name and ID              */
   long      lBeginBaseTime;          /* Beginning Model Basetime           */
   long      lEndBaseTime;            /* Ending Model Basetime              */
   long      lBeginTau;               /* Begin Forecast Period  (minutes)   */
   long      lEndTau;                 /* Ending Forecast Period (minutes)   */
   long      lBeginReceiptTime;       /* Begin Receipt Time                 */
   long      lEndReceiptTime;         /* End   Receipt Time                 */
} MAGRIDDELETEQUERY, *PMAGRIDDELETEQUERY;
```

3.5 MAGRID Error Definitions

The following are the MAGRID error definitions contained in the file `MAGRIDErr.h`.

```
#ifndef    MAGRIDERR_OFFSET
#define    MAGRIDERR_OFFSET    9000
#endif
#define    MAGRID_TEDSALLOC                  1 + MAGRIDERR_OFFSET
#define    MAGRID_DATANOTFOUND               2 + MAGRIDERR_OFFSET
#define    MAGRID_TEDSNULLQUAL               3 + MAGRIDERR_OFFSET
#define    MAGRID_NULL_PTR                   4 + MAGRIDERR_OFFSET
#define    MAGRID_INVALIDDATATYPE            5 + MAGRIDERR_
#define    MAGRID_INVALID_MODEL              6 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_UNIT               7 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_TAU                8 + MAGRIDERR_OFFSET
#define    MAGRID_NAMETOLONG                 9 + MAGRIDERR_OFFSET
#define    MAGRID_CONNECTION_NAME_REQUIRED   10 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_SUBCENTER          11 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_PRODUCTION_CENTER  12 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_PARAMETER          13 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_BASETIME           14 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_AREA               15 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_RECORDID           16 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_DATASET_NAME       17 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_PROJECTION         18 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_DIMENSIONS         19 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_UNITID             20 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_SCANMODE           21 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_REGISTRATION       22 + MAGRIDERR_OFFSET
#define    MAGRID_UNKNOWN_REGISTRATION       23 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_TYPE_FOR_DATASET   24 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_GRIDID             25 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_LEVEL              26 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_QUALITY_INDICATOR  27 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_DATA_CATEGORY      28 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_LATITUDE           29 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_LONGITUDE          30 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_XPOINTS            31 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_YPOINTS            32 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_REGX               33 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_REGY               34 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_XDISTANCE          35 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_YDISTANCE          36 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_STANDARD_LAT       37 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_STANDARD_LON       38 + MAGRIDERR_OFFSET
#define    MAGRID_UNKNOWN_PARM_ID            39 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_LEVELTYPE_ID       40 + MAGRIDERR_OFFSET
```

```
#define    MAGRID_INVALID_CONVERSION_REQUEST     41 + MAGRIDERR_OFFSET
#define    MAGRID_REGISTRATION_MISMATCH          42 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_LOCK_MODE              43 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_ROLE_NAME              44 + MAGRIDERR_OFFSET
#define    MAGRID_MUST_SPECIFY_CENTER            45 + MAGRIDERR_OFFSET
#define    MAGRID_MUST_SPECIFY_SUBCENTER         46 + MAGRIDERR_OFFSET
#define    MAGRID_GRIDID_NOT_AVAILABLE           47 + MAGRIDERR_OFFSET
#define    MAGRID_LEVELS_MISMATCH                48 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_3DREG_ID               49 + MAGRIDERR_OFFSET
#define    MAGRID_AOI_CALCULATION_FAILED         50 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_MAXZPOINT              51 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_TRACK_RANGE            52 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_TRACK_BEARING          53 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_TRACK_RESOLUTION       54 + MAGRIDERR_OFFSET
#define    MAGRID_NO_LEVEL_VALUE_MATCH           55 + MAGRIDERR_OFFSET
#define    MAGRID_BLOB_SIZE_MISMATCH             56 + MAGRIDERR_OFFSET
#define    MAGRID_INVALID_GRID_SIZE              57 + MAGRIDERR_OFFSET
#define    MAGRID_DROP_TABLE_FAILED              58 + MAGRIDERR_OFFSET
#define    MAGRID_DYNAMIC_CLEANUP_FAILED         59 + MAGRIDERR_OFFSET
#define    MAGRID_PROJECTION_ALREADY_APPLIED     60 + MAGRIDERR_OFFSET
#define    MAGRID_MODEL_IN_USE                   61 + MAGRIDERR_OFFSET
#define    MAGRID_CENTER_IN_USE                  62 + MAGRIDERR_OFFSET
#define    MAGRID_PARAMETER_IN_USE               63 + MAGRIDERR_OFFSET
#define    MAGRID_REGISTRATION_IN_USE            64 + MAGRIDERR_OFFSET
#define    MAGRID_ID_EXISTS                      65 + MAGRIDERR_OFFSET
#define    MAGRID_NO_CENTER                      66 + MAGRIDERR_OFFSET
#define    MAGRID_EOF_AS_STORED                  67 + MAGRIDERR_OFFSET
```

(This page intentionally left blank.)

4 CONNECT APIS

The connect APIs are used to establish or disestablish a connection to the database server on which the MDGRID database resides.

Information about each API is presented in manual page format as follows:

NAME
Function Name – Provides a brief description of the function.

SYNOPSIS
Presents the calling syntax for the routine, including the declarations of the arguments and the return type. Also lists the necessary include files for each routine.

INPUT PARAMETERS
Describes each of the input parameters used by the function.

OUTPUT PARAMETERS
Describes each of the parameters output by the function.

DESCRIPTION
Describes what the function does and what events or side effects it causes.

RETURNS
Describes what the function returns.

NOTE
Provides any applicable notes about the function.

SEE ALSO
Provides a reference to related functions.

Examples showing the proper usage of the APIs are presented in the *Grid Field API Programming Manual*, referenced in Section 2.

4.1 MAGRIDConnect

NAME
`MAGRIDConnect` – Connects the application to the database.

SYNOPSIS
```
#include "MAGRIDAPI.h"
MAGRIDRET MAGRIDConnect(void);
```

INPUT PARAMETERS
None.

OUTPUT PARAMETERS
None.

DESCRIPTION
This subroutine connects the application to the database.

RETURNS
MAGRIDRET structure – See Section 3.4.18 for details.

NOTE
1. This interface must be called before any other MAGRID APIs may be called. It should only be called once per session.

SEE ALSO
MAGRIDDisconnect, MAGRIDRemoteConnect

4.2 MAGRIDDisconnect

NAME
`MAGRIDDisconnect` – Disconnects the application from the database.

SYNOPSIS
```
#include “MAGRIDAPI.h”
MAGRIDRET MAGRIDDisconnect(void);
```

INPUT PARAMETERS
None.

OUTPUT PARAMETERS
None.

DESCRIPTION
This subroutine disconnects the application from the database, ending the database session.

RETURNS
MAGRIDRET structure – See Section 3.4.18 for details.

NOTE
1. MAGRIDDisconnect should be called to end each database session started by MAGRIDConnect. It should be called once per session.

SEE ALSO
MAGRIDConnect

4.3 MAGRIDRemoteConnect

NAME

`MAGRIDRemoteConnect`	-	Allows the application to connect to a database residing on a remote server and/or to establish multiple connections.

SYNOPSIS

```
#include “MAGRIDAPI.h”
MAGRIDRET MAGRIDConnect
(
      char *szDBServer,
      char *szConnectionName
);
```

INPUT PARAMETERS

char *szDBServer	-	The database server name (COE default is $INFORMIXSERVER)
char *szConnectionName	-	The connection name for mdgrid database MAGRID_DEFAULT_CONNECTION is default)

OUTPUT PARAMETERS

None.

DESCRIPTION

The MAGRIDRemoteConnect routine allows an application to connect to a remote database server and/or to maintain multiple open connections on the same or different servers.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTES

1. This interface must be called before any other MAGRID APIs may be called. It should only be called once per session with the same arguments.
2. Passing in NULL for either argument assumes that the default settings for server ($INFORMIXSERVER) and connection name (`MAGRID_DEFAULT_CONNECTION)`.
3. Applications connecting to only one server and one database at a time should use the simpler MAGRIDConnect and MAGRIDDisconnect routines.

SEE ALSO

MAGRIDRemoteDisconnect, MAGRIDSetConnection, MAGRIDConnect

4.4 MAGRIDRemoteDisconnect

NAME

`MAGRIDRemoteDisconnect`	-	Disconnects the application from the database residing on a remote server.

SYNOPSIS
```
#include "MAGRIDAPI.h"
MAGRIDRET MAGRIDRemoteDisconnect
(
char *szConnectionName
);
```

INPUT PARAMETERS

char *szConnectionName	-	The connection to be disconnected.

OUTPUT PARAMETERS
None.

DESCRIPTION
This subroutine disconnects the application from the database connection specified, ending the database session.

RETURNS
MAGRIDRET structure – See Section 3.4.18 for details.

NOTE
1. MAGRIDRemoteDisconnect should be called to end each database connection session started by MAGRIDRemoteConnect. It should only be called once per session with the same arguments.

SEE ALSO
MAGRIDRemoteConnect, MAGRIDSetConnection

4.5 MAGRIDSetConnection

NAME

`MAGRIDSetConnection` - Sets the connection context for multiple connections to different database servers and/or databases.

SYNOPSIS

```
#include “MAGRIDAPI.h”
MAGRIDRET MAGRIDSetDisconnect
(
char *szConnectionName
);
```

INPUT PARAMETERS

char *szConnectionName - The connection desired for database transactions.

OUTPUT PARAMETERS

None.

DESCRIPTION

The MAGRIDSetConnect routine is used to set the connection context for multiple connections to different database servers and/or databases.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTE

1. Passing in NULL for the connection name assumes the default connection is to be used `(MAGRID_DEFAULT_CONNECTION)`.

SEE ALSO

MAGRIDRemoteConnect, MAGRIDRemoteDisconnect

5 RETRIEVAL APIS

This section describes the APIs that are used to retrieve data from the MDGRID database.

5.1 MAGRIDRetrRegistration

NAME
`MAGRIDRetrRegistration` – Retrieves a linked list of registration information.

SYNOPSIS
```
#include "MAGRIDAPI.h"
MAGRIDRET MAGRIDRetrRegistration
(
      long                    lGridId,
      long                    lCenterId,
      long                    lSubCenterId,
      PMAGRIDLINKEDLIST       pRegistrationLL
)
```

INPUT PARAMETERS

long lGridId	– A Grid ID
long lCenterId	– A Center ID
long lSubCenterId	– A Sub Center ID

OUTPUT PARAMETERS

PMAGRIDLINKEDLIST pRegistrationLL	– A pointer to a MAGRIDLINKEDLIST structure (Section 3.4.17) that points to the MAGRIDOUTPUTREG structure (Section 3.4.19) containing the retrieved registration information.

DESCRIPTION
The MAGRIDRetrRegistration routine retrieves registration information from the database given the gridID, centerID, and subcenterID. It returns a linked list that points to MAGRIDOUTPUTREG structures containing the return registration information. It also returns a MAGRIDRET structure indicating the status of the operation.

RETURNS
MAGRIDRET structure – See Section 3.4.18 for details.

NOTES
1. MAGRIDConnect must have been called to start a database session before MAGRIDRetrRegistration may be called.
2. The gridID, centerID, and subcenterID may be wildcarded.

3. MAGRIDFreeLL should be called to free the linked list upon completion of processing.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect MAGRIDFreeLL, MAGRIDVerifyModel, MAGRIDRegisterModel

5.2 MAGRID2DCatalog

NAME
MAGRID2DCatalog – Retrieves a catalog listing of 2D Grid Fields meeting specified criteria.

SYNOPSIS
```
#include "MAGRIDAPI.h"
 MAGRIDRET MAGRID2DCatalog
 (
       PMAGRIDQUERY pGridQuery,
       long *lNumFound,
       PMAGRIDLINKEDLIST pCatList
);
```

INPUT PARAMETERS

PMAGRIDQUERY pGridQuery	–	A pointer to a MAGRIDQUERY structure (Section 3.4.7) containing query criteria for the catalog retrieval.

OUTPUT PARAMETERS

long *lNumFound	–	A pointer to the number of records found that match the query criteria.
PMAGRIDLINKEDLIST pCatList	–	A pointer to the MAGRIDLINKEDLIST structure (Section 3.4.17), which points to the MAGRIDFIELDDESC (Section 3.4.10) structures containing the catalog data retrieved.

DESCRIPTION
The MAGRID2DCatalog function retrieves a catalog listing of grids in the database that satisfy the query criteria provided in the input MAGRIDQUERY structure. It returns the number of catalog items found, a linked list of the catalog items, and the return status structure, MAGRIDRET.

RETURNS
MAGRIDRET structure – See Section 3.4.18 for details.

NOTES
1. The grid query structure should be initialized to zero.
2. MAGRIDFreeLL should be called to free the linked list upon completion of processing.
3. The AOI (pGridQuery->stGeoArea) must be specified – no wildcards. All other fields of the grid query structure may be wildcarded with the following exception. Site-specific parameters range between 128-254, according to the GRIB Specification. If the lParameter is

set to a value between 128 and 254, then the lProductionCenterId and lSubCenterId cannot be wildcarded. While wildcarding is permitted and maximum flexibility is allowed, the programmer should use some discretion when using them. Wildcarding is likely to result in a large volume of data being returned, which may tax system/network resources.

4. The rsEndLevel is intended for use when the level type is a layer. If the user is interested in level type that represents a single level, then rsEndLevel should be wildcarded and rsBeginLevel should be set to the level of interest or may be wildcarded to get all levels. If the user is interested in a level type that represents a layer, then both rsBeginLevel and rsEndLevel should be set.
5. MAGRIDConnect must have been called to start a database session before MAGRID2DCatalog may be called.
6. If the Data Category field is not explicitly set to wildcard, then only the base grids will be retrieved. A base grid is one that was ingested from the decoders. An edited grid is one that was updated via the MAGRIDUpdateByID API. A derived grid is one that was retrieved from the database, registration applied, and then saved back to the database.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect, MAGRIDFreeLL

5.3 MAGRID3DCatalog

NAME

`MAGRID3DCatalog` – Retrieves a catalog listing of 3D Grid Fields meeting specified criteria.

SYNOPSIS

```
#include "MAGRIDAPI.h"
 MAGRIDRET MAGRID3DCatalog
 (
       PMAGRID3DQUERY pGridQuery,
       long *lNumFound,
       PMAGRIDLINKEDLIST pCatList
);
```

INPUT PARAMETERS

PMAGRID3DQUERY pGridQuery	–	A pointer to a MAGRID3DQUERY structure (Section 3.4.8) containing query criteria for the catalog retrieval.

OUTPUT PARAMETERS

long *lNumFound	–	A pointer to the number of records found that match the query criteria.
PMAGRIDLINKEDLIST pCatList	–	A pointer to the MAGRIDLINKEDLIST structure (Section 3.4.17), which points to the MAGRID3DFIELDDESC (Section 3.4.11) structures containing the catalog data retrieved.

DESCRIPTION

The MAGRID3DCatalog function retrieves a catalog listing of 3D grids in the database that satisfy the query criteria provided in the input MAGRID3DQUERY structure. It returns the number of catalog items found, a linked list of the catalog items, and the return status structure, MAGRIDRET.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTES

1. The grid query structure should be initialized to zero.
2. MAGRIDFreeLL should be called to free the linked list upon completion of processing.
3. The AOI (pGridQuery->stGeoArea) must be specified – no wildcards. All other fields of the grid query structure may be wildcarded with the following exception. Site-specific parameters range between 128-254, according to the GRIB Specification. If the lParameter is

set to a value between 128 and 254, then the lProductionCenterId and lSubCenterId cannot be wildcarded. While wildcarding is permitted and maximum flexibility is allowed, the programmer should use some discretion when using them. Wildcarding is likely to result in a large volume of data being returned, which may tax system/network resources.

4. The rsEndLevel is intended for use when the level type is a layer. If the user is interested in level type that represents a single level, then rsEndLevel should be wildcarded and rsBeginLevel should be set to the level of interest or may be wildcarded to get all levels. If the user is interested in a level type that represents a layer, then both rsBeginLevel and rsEndLevel should be set.
5. MAGRIDConnect must have been called to start a database session before MAGRID3DCatalog may be called.
6. If the Data Category field is not explicitly set to wildcard, then only the base grids will be retrieved. A base grid is one that was ingested from the decoders. An edited grid is one that was updated via the MAGRIDUpdateByID API. A derived grid is one that was retrieved from the database, registration applied, and then saved back to the database.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect, MAGRIDFreeLL

5.4 MAGRIDGet2DByID

NAME

`MAGRIDGet2DByID`	–	Retrieves a single 2D grid from the database given the datasetname, recordId, and desired format of the data.

SYNOPSIS

```
#include "MAGRIDAPI.h"
 MAGRIDRET  MAGRIDGet2DByID
 (
      PMAGRIDREFERENCE pGridReference,
      PMAGRIDDATA pGridData
);
```

INPUT PARAMETERS

PMAGRIDREFERENCE pGridReference	-	A pointer to the MAGRIDREFERENCE structure (Section 3.4.12) containing the description of the grid to be retrieved.

OUTPUT PARAMETERS

PMAGRIDDATA pGridData	-	An address pointer that points to the MAGRIDDATA structure (Section 3.4.14) containing the retrieved grid.

DESCRIPTION

The MAGRIDGet2DByID routine takes as input a pointer to a MAGRIDREFERENCE structure, which contains a description of the grid to be retrieved. The datasetname and recordId uniquely identify a grid in the database. The remainder of the MAGRIDREFERENCE fields describe the desired format of the data (e.g., projection, units). If pGridReference $\rightarrow$ StGridFormat.eOutputFormat is set to MAGRID_GET_AS_SPECIFIED, the grid will be retrieved as specified in the GRIDREFERENCE structure. If pGridReference $\rightarrow$ StGridFormat.eOutputFormat is set to MAGRID_GET_AS_STORED, the grid will be retrieved as stored in the database. No subgridding, projections, or registration will be applied to the grid field data. When data is retrieved this way, a corresponding call to retrieve the registration information will most likely be required. It returns the requested grid in the MAGRIDDATA structure.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTES

1. MAGRIDConnect must have been called to start a database session before MAGRIDGet2DByID may be called.

2. If pGridReference → stGridFormat.eOutputFormat=MAGRID_GET_AS_SPECIFIED, then all fields in the MAGRIDREFERENCE structure must be set to a value; wildcards are not permitted.
3. If pGridReference → stGridFormat.eOutputFormat=MAGRID_GET_AS_STORED, then only the DatasetName and lRecordId need to be set.
4. Missing values in the retrieved grid are represented as -9999.
5. The following structure is only applicable for polar or lambert projection requests: pGridReference → stGridFormat.stProjectionDesc.projParms. This structure is used to specify the standard latitudes and longitudes for a polar or lambert retrieval. In the case of a spherical or mercator retrieval, this structure is ignored.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect, MAGRID2DCatalog

5.5 MAGRIDGet2DByQuery

NAME

`MAGRIDGet2DByQuery`	–	Retrieves one or more grids from the database that match the specified query criteria.

SYNOPSIS

```
include "MAGRIDAPI.h"
 MAGRIDRET  MAGRIDGet2DByQuery
 (
      PMAGRIDQUERY pGridQuery,
      PMAGRIDFORMAT pGridFormat,
      long *lNumFound,
      PMAGRIDLINKEDLIST pGridDataLL
);
```

INPUT PARAMETERS

PMAGRIDQUERY pGridQuery	–	A pointer to a MAGRIDQUERY structure (Section 3.4.8) containing query criteria for the catalog retrieval.
PMAGRIDFORMAT pGridFormat	–	A pointer to the format specification structure of the returned grids.

OUTPUT PARAMETERS

long *lNumFound	–	A pointer to the number of 2D grids in the linked list.
PMAGRIDLINKEDLIST pGridDataLL	–	A pointer to the MAGRIDLINKEDLIST structure (Section 3.4.17), which points to the MAGRIDDATA structure (Section 3.4.14) containing the retrieved grids.

DESCRIPTION

The MAGRIDGet2DByQuery routine takes as input a pointer to a MAGRIDQUERY structure containing the query criteria. It returns the number of matching grids found, a linked list that points to MAGRIDDATA structures containing the grids, and the MAGRIDRET return status structure.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTES

1. The grid query structure should be initialized to zero.
2. MAGRIDFreeLL should be called to free the linked list upon completion of processing.
3. The AOI (pGridQuery->stGeoArea) must be specified – no wildcards. All other fields of the grid query structure may be wildcarded with the following exception. Site-specific parameters range between 128-254, according to the GRIB Specification. If the lParameter is set to a value between 128 and 254, then the lProductionCenterId and lSubCenterId cannot be wildcarded. While wildcarding is permitted and maximum flexibility is allowed, the programmer should use some discretion when using them. Wildcarding is likely to result in a large volume of data being returned, which may tax system/network resources.
4. The rsEndLevel is intended for use when the level type is a layer. If the user is interested in level type that represents a single level, then rsEndLevel should be wildcarded and rsBeginLevel should be set to the level of interest or may be wildcarded to get all levels. If the user is interested in a level type that represents a layer, then both rsBeginLevel and rsEndLevel should be set.
5. MAGRIDConnect must have been called to start a database session before MAGRID2DByQuery may be called.
6. If the Data Category field is not explicitly set to wildcard, then only the base grids will be retrieved. A base grid is one that was ingested from the decoders. An edited grid is one that was updated via the MAGRIDUpdateByID API. A derived grid is one that was retrieved from the database, registration applied, and then saved back to the database.
7. If pGridFormat →eOutputFormat=MAGRID_GET_AS_SPECIFIED, then all fields of the MAGRIDFORMAT must be set - no wildcards.
8. MAGRIDFreeLL should be called to free the linked list upon completion of processing.
9. Missing values in the retrieved grid are represented as -9999.
10. The following structure is only applicable for polar or lambert projection requests: pGridFormat → stProjectionDesc.projParms. This structure is used to specify the standard latitudes and longitudes for a polar or lambert retrieval. In the case of a spherical or mercator retrieval, this structure is ignored.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect, MAGRIDFreeLL

5.6 MAGRIDGetVolumeByID

NAME

`MAGRIDGetVolumeByID` – Retrieves a single 3D grid from the database given the datasetname, recordId, and desired format of the data.

SYNOPSIS

```
include "MAGRIDAPI.h"
 MAGRIDRET  MAGRIDGetVolumeByID
 (
      PMAGRIDREFERENCE pGridReference,
      PMAGRID3DDATA pGridData
);
```

INPUT PARAMETERS

PMAGRIDREFERENCE pGridReference	-	A pointer to the MAGRIDREFERENCE structure (Section 3.4.12) containing the description of the grid to be retrieved.

OUTPUT PARAMETERS

PMAGRID3DDATA pGridData	-	A pointer to the MAGRID3DDATA structure (Section 3.4.15) containing the retrieved grid.

DESCRIPTION

The MAGRIDGetVolumeByID routine takes as input a pointer to a MAGRIDREFERENCE structure, which contains a description of the grid to be retrieved. The datasetname and recordId uniquely identify a grid in the database. The remainder of the MAGRIDREFERENCE fields describe the desired format of the data (e.g., projection, units). It returns the requested grid in the MAGRID3DDATA structure.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTES

1. MAGRIDConnect must have been called to start a database session before MAGRIDGetVolumeByID may be called.
2. If pGridReference → StGridFormat.eOutputFormat=MAGRID_GET_AS_SPECIFIED, then all fields in the MAGRIDREFERENCE structure must be set to a value; wildcards are not permitted.
3. If pGridReference →StGridFormat.eOutputFormat=MAGRID_GET_AS_STORED, then only the DatasetName and lRecordId need to be set.
4. Missing values in the retrieved grid are represented as -9999.
5. The following structure is only applicable for polar or lambert projection requests: pGridReference → stGridFormat.stProjectionDesc.projParms. This structure is used to

specify the standard latitudes and longitudes for a polar or lambert retrieval. In the case of a spherical or mercator retrieval, this structure is ignored.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect, MAGRID3DCatalog

5.7 MAGRIDGetSliceByID

NAME

`MAGRIDGetSliceByID`	–	Retrieves a single 3D grid horizontal slice (2D Grid at a designated level) from the database given the datasetname, recordId, and desired format of the data.

SYNOPSIS

```
include "MAGRIDAPI.h"
 MAGRIDRET  MAGRIDGetSliceByID
 (
      PMAGRIDREFERENCE pGridReference,
      float rsLevel,
      PMAGRIDDATA pGridData
);
```

INPUT PARAMETERS

PMAGRIDREFERENCE pGridReference	-	A pointer to the MAGRIDREFERENCE structure (Section 3.4.12) containing the description of the grid to be retrieved.
float rsLevel	-	A level value indicator.

OUTPUT PARAMETERS

PMAGRIDDATA pGridData	-	A pointer that points to the MAGRIDDATA structure (Section 3.4.14 containing the retrieved grid.

DESCRIPTION

The MAGRIDGetSliceByID routine takes as input a pointer to a MAGRIDREFERENCE structure and a float value, which is the level of interest. The 2D grid returned is a horizontal slice of the designated 3D grid. The datasetname and recordId uniquely identify a 3D grid in the database. The remainder of the MAGRIDREFERENCE fields describe the desired format of the data (e.g., projection, units). It returns the requested horizontal slice in the MAGRIDDATA structure.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTES

1. MAGRIDConnect must have been called to start a database session before MAGRIDGetSliceByID may be called.

2. If pGridReference →StGridFormat.eOutputFormat=MAGRID_GET_AS_SPECIFIED, then all fields in the MAGRIDREFERENCE structure must be set to a value; wildcards are not permitted.
3. If pGridReference →StGridFormat.eOutputFormat=MAGRID_GET_AS_STORED, then only the DatasetName and lRecordId need to be set.
4. Missing values in the retrieved grid are represented as -9999.
5. The following structure is only applicable for polar or lambert projection requests: pGridReference → stGridFormat.stProjectionDesc.projParms. This structure is used to specify the standard latitudes and longitudes for a polar or lambert retrieval. In the case of a spherical or mercator retrieval, this structure is ignored.

SEE ALSO
MAGRIDConnect, MAGRIDRemoteConnect, MAGRID3DCatalog

5.8 MAGRIDGetProfileByID

NAME

`MAGRIDGetProfileByID` – Retrieves a 3D grid profile (stick) from the database given the datasetname and recordId.

SYNOPSIS

```
include "MAGRIDAPI.h"
 MAGRIDRET  MAGRIDGetProfileByID
 (
      char *szDataSetName,
      long lRecordID,
      float rsLat,
      float rsLong,
      PMAGRID3DSTICK pProfileData
);
```

INPUT PARAMETERS

char *szDataSetName	–	The name of the dataset from which the record is to be deleted.
long lRecordID	–	An identifier for the record that is to be deleted from the named dataset.
float rsLat	–	A profile latitude position
float rsLong	–	A profile longitude position

OUTPUT PARAMETERS

PMAGRID3DSTICK pProfileData	–	A pointer to a MAGRID3DSTICK data structure (Section 3.4.16) containing the profile data for the 3D stick requested.

DESCRIPTION

The MAGRIDGetProfileByID routine takes as input the datasetname and recordId, which uniquely identifies a grid in the database, and the latitude/longitude for point of interest. It returns the requested profile in the PMAGRID3DSTICK data structure.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTE

1. MAGRIDConnect must have been called to start a database session before MAGRIDGetProfileByID may be called.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect, MAGRID3DCatalog

5.9 MAGRIDGetTrack

NAME

`MAGRIDGetTrack`	–	Retrieves a track of profile points given the starting lat/long position, range, bearing, and resolution.

SYNOPSIS
```
include "MAGRIDAPI.h"
 MAGRIDRET MAGRIDGetTrack
 (
      MAGRID3DTRACKQUERY TrackSpecification,
      long *pNumberOfTrackPts,
      PMAGRIDLINKEDLIST pTrackData
);
```

INPUT PARAMETERS

MAGRID3DTRACKQUERY TrackSpecification	–	MAGRID3DTRACKQUERY data structure (Section 3.4.9) containing a description of the track data to be extracted.

OUTPUT PARAMETERS

long *pNumberOfTrackPts	–	A pointer to the number of track points extracted that meet the criteria described by the request.
PMAGRIDLINKEDLIST pTrackData	–	A pointer to a MAGRIDLINKEDLIST (Section 3.4.17) of MAGRID3DSTICK (Section 3.4.16) profiles.

DESCRIPTION

The MAGRIDGetTrack routine takes as input a MAGRID3DTRACKQUERY structure, which contains a description of the track data to be retrieved. It returns the requested track of profiles in the MAGRIDLINKEDLIST of MAGRID3DSTICK profiles . It also returns the number of track points retrieved that match the MAGRID3DTRACKQUERY specification.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTE

1. MAGRIDConnect must have been called to start a database session before MAGRIDGetTrack may be called.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect

5.10 MAGRIDGetModelsInfo

NAME

`MAGRIDGetModelsInfo`	–	Retrieves the list of model (generating process) information based on the query specified.

SYNOPSIS

```
#include “MAGRIDAPI.h”
MAGRIDRET MAGRIDGetModelsInfo
(
      long lCenterId,
      long lSubCenterId,
      long lGeneratingProcId,
      long lGridId,
      long *plNumFound,
      PMAGRIDLINKEDLIST pModelInfoLL
);
```

INPUT PARAMETERS

long lCenterId	–	Center ID
long lSubCenterId	–	Sub Center ID
long lGeneratingProcId	–	Generating Process ID (model ID)
long lGridId	–	Grid ID

OUTPUT PARAMETERS

long *plNumFound	–	A pointer to the number of center descriptive records found that meet the selection criteria.
PMAGRIDLINKEDLIST pModelInfoLL	–	A pointer to a MAGRIDLINKEDLIST structure (Section 3.4.17), which points to the MAGRIDMODELDATA structure (Section 3.4.21) containing the retrieved model information.

DESCRIPTION

This subroutine retrieves the list of model records found to match the center/subcenter ids, model id, and grid id specified (any of which may be wildcarded). The information retrieved in each record is the model id, model name, center id, subcenter id, grid id, and center name.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTES

1. MAGRIDConnect must have been called to start a database session before MAGRIDGetModelsInfo may be called.
2. lCenterId, lSubCenterId lGeneratingProcId, and lGridId may be wildcarded by using MAGRID_QUERY_WILDCARD. When all four are wildcarded, the retrieval will return the all model definitions known to the system. A setting of all four with specific values will return one record as found for the specified center/subcenter/generatingproc/grid id combination.
3. MAGRIDFreeLL should be called to free the linked list upon completion of processing.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect

5.11 MAGRIDGetParametersInfo

NAME

`MAGRIDGetParametersInfo`	– Retrieves the list of parameter (element) information based on the query specified.

SYNOPSIS

```
#include “MAGRIDAPI.h”
MAGRIDRET MAGRIDGetParametersInfo
(
      long lCenterID,
      long lSubCenterId,
      long lParameterId,
      long *plNumFound,
      PMAGRIDLINKEDLIST pParameterInfoLL
);
```

INPUT PARAMETERS

long lCenterId	– Center ID
long lSubCenterId	– Sub Center ID
long lParameterId	– Parameter ID

OUTPUT PARAMETERS

long *plNumFound	– A pointer to the number of parameter descriptive records found that meet the selection criteria.
PMAGRIDLINKEDLIST pParameterInfoLL	– A pointer to a MAGRIDLINKEDLIST structure (Section 3.4.17), which points to the MAGRIDPARAMETERDATA structure (Section 3.4.22) containing the retrieved Parameter information.

DESCRIPTION

This subroutine retrieves the list of parameter records found to match the centers/subcenter and parameter specified (any of which may be wildcarded). The information retrieved in each record is the parameter id, parameter name, default unit id, min unit value, max unit value, default unit name, and default unit abbreviation. Center id, subcenter id fields are also returned where appropriate. Parameter ids that fall within the range 1-127 are considered common parameters. Common parameters apply to all centers/subcenters. Parameter Ids that are greater than 127 are center-specific parameters. When a common parameter record is returned, the center and subcenter id are set to the constant MAGRID_COMMON_PARAMETER.

RETURNS
MAGRIDRET structure – See Section 3.4.18 for details.

NOTES
1. MAGRIDConnect must have been called to start a database session before MAGRIDGetParametersInfo may be called.
2. lCenterID, lSubCenterID, and lParameterID may be wildcarded by using MAGRID_QUERY_WILDCARD. When all three are wildcarded, the retrieval will return all the parameter definitions known to the system (common and center/subcenter-specific parameters). A setting of all three with specific values will return one record as found for the specified center/subcenter/parameter id combination.
3. MAGRIDFreeLL should be called to free the linked list upon completion of processing.

SEE ALSO
MAGRIDConnect, MAGRIDRemoteConnect

5.12 MAGRIDGetUnitsInfo

NAME

`MAGRIDGetUnitsInfo` – Retrieves the list of unit information based on the query specified.

SYNOPSIS

```
#include "MAGRIDAPI.h"
MAGRIDRET MAGRIDGetUnitsInfo
(
      long lUnitId,
      long *plNumFound,
      PMAGRIDLINKEDLIST pUnitInfoLL
);
```

INPUT PARAMETERS

long lUnitId	–	Unit ID

OUTPUT PARAMETERS

long *plNumFound	–	A pointer to the number of unit descriptive records found that meet the selection criteria.
PMAGRIDLINKEDLIST pUnitInfoLL	–	A pointer to a MAGRIDLINKEDLIST structure (Section 3.4.17), which points to the MAGRIDUNITDATA structure (Section 3.4.23) containing the retrieved unit information.

DESCRIPTION

This subroutine retrieves the list of unit records found to match the unit id (which may be wildcarded). The information retrieved in each record is the unit id, unit abbreviation, and unit name fields.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTES

1. MAGRIDConnect must have been called to start a database session before MAGRIDGetUnitsInfo may be called.
2. The lUnitId may be wildcarded by using MAGRID_QUERY_WILDCARD. If the lUnitId field is wildcarded, the retrieval will return all the unit definitions known to the database. A setting of a specific value will return one record as found for the specified unit id.
3. MAGRIDFreeLL should be called to free the linked list upon completion of processing.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnection

5.13 MAGRIDGetCentersInfo

NAME

`MAGRIDGetCentersInfo` – Retrieves the list of center information based on query specified.

SYNOPSIS

```
#include “MAGRIDAPI.h”
MAGRIDRET MAGRIDGetCentersInfo
(
      long lCenterID,
      long lSubCenterId,
      long *plNumFound,
      PMAGRIDLINKEDLIST pUnitCenterLL
);
```

INPUT PARAMETERS

long lCenterId	–	Center ID
long lSubCenterId	–	Sub Center ID

OUTPUT PARAMETERS

long *plNumFound	–	A pointer to the number of center descriptive records found that meet the selection criteria.
PMAGRIDLINKEDLIST pUnitInfoLL	–	A pointer to a MAGRIDLINKEDLIST structure (Section 3.4.17), which points to the MAGRIDCENTERDATA structure (Section 3.4.20) containing the retrieved center information.

DESCRIPTION

This subroutine retrieves the list of center records found to match the center/subcenter id specified (may be wildcarded). The information retrieved in each record is the centerid, subcenterid, and center name fields.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTES

1. MAGRIDConnect must have been called to start a database session before MAGRIDGetCentersInfo may be called.
2. The lCenterId and lSubCenterId may be wildcarded by using MAGRID_QUERY_WILDCARD. Wildcarding both fields will cause the retrieval of all the

center definitions known to the database. A setting of a specific value will return one record as found for the specified center/subcenter id.

3. MAGRIDFreeLL should be called to free the linked list upon completion of processing.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnection

(This page intentionally left blank.)

6 DATA MANAGEMENT APIS

6.1 MAGRID2DIngest

NAME

`MAGRID2DIngest` – Ingests a 2D Grid Field record to the database.

SYNOPSIS

```
#include "MAGRIDAPI.h"
MAGRIDRET MAGRID2DIngest
(
PMAGRID2DINGEST   pMAGRIDData
);
```

INPUT PARAMETERS

PMAGRID2DINGEST pMAGRIDData	–	A pointer to a MAGRID2DINGEST structure (Section 3.4.3) containing Grid Field description and data to be ingested.

OUTPUT PARAMETERS

PMAGRID2DINGEST pMAGRIDData	–	A pointer to a MAGRID2DINGEST structure (Section 3.4.3) containing updated reference identifiers (szDataSetName, lRecordId).

DESCRIPTION

The MAGRID2DIngest function ingests a Grid Field into the database. The routine takes as input a pointer to the MAGRID2DINGEST structure containing the grid field description and data. The routine returns the reference identifiers (szDataSetName, lRecordId) after the grid has been added to the database.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTES

1. This interface supersedes the MAGRIDStore interface provided in the developer release. All applications using MAGRIDStore should be modified to call MAGRID2DIngest instead, as MAGRIDStore will be phased out in the next release.
2. MAGRIDConnect must have been called to start a database session before MAGRID2DIngest may be called.
3. If the unit ID field in the MAGRID2DIngest structure is not set (equal 0), the default value that is associated with the parameter ID will be used.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect

6.2 MAGRID3DIngest

NAME

`MAGRID3DIngest` – Ingests a 3D Grid Field record into the database.

SYNOPSIS

```
#include “MAGRIDAPI.h”
MAGRIDRET MAGRID3DIngest
(
PMAGRID3DINGEST   pMAGRIDData
);
```

INPUT PARAMETERS

PMAGRID3DINGEST pMAGRIDData	–	A pointer to a MAGRID3DINGEST structure (Section 3.4.4) containing 3D Grid Field description and data to be ingested.

OUTPUT PARAMETERS

PMAGRID3DINGEST pMAGRIDData	–	A pointer to a MAGRID3DINGEST structure (Section 3.4.4) containing updated reference identifiers (szDataSetName, lRecordId).

DESCRIPTION

The MAGRID3DIngest function ingests a 3D Grid Field into the database. The routine takes as input a pointer to the MAGRID3DINGEST structure containing the 3D grid field description and data. The routine returns the reference identifiers (szDataSetName, lRecordId) after the 3D grid has been added to the database.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTES

1. MAGRIDConnect must have been called to start a database session before MAGRID3DIngest may be called.
2. If the unit ID field in the MAGRID3DINGEST structure is not set (equal 0), the default value that is associated with the parameter ID will be used.
3. This API may be used to ingest a 3D EOF grid. The pMAGRIDData->szCompression should be set to MAGRID_EOF_COMPRESSION. See the Programming Manual for further details.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect

6.3 MAGRIDDeleteByID

NAME

`MAGRIDDeleteByID` – Deletes a record (2D or 3D) from a dataset in the database.

SYNOPSIS

```
include "MAGRIDAPI.h"
 MAGRIDRET  MAGRIDDeleteByID
 (
      char *szDataSetName,
      long lRecordID
);
```

INPUT PARAMETERS

char *szDataSetName	–	The name of the dataset from which the record is to be deleted.
long lRecordID	–	An identifier for the record that is to be deleted from the named dataset.

OUTPUT PARAMETERS

None.

DESCRIPTION

The MAGRIDDeleteByID routine deletes a record from a dataset, given the name of the dataset and the ID of the record. It returns a MAGRIDRET structure indicating the status of the operation.

RETURNS

MAGRIDRET Structure – See Section 3.4.18 for details.

NOTES

1. Typically a call to MAGRID2DCatalog or MAGRID3DCatalog to get a list of grids is made before the MAGRIDDeleteByID call.
2. MAGRIDConnect must have been called to start a database session before MAGRIDDeleteByID may be called.
3. A Dataset Table can only be deleted by either the user who created it or by a user given dba permission.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect, MAGRID2DCatalog, MAGRID3DCatalog

6.4 MAGRIDDeleteByQuery

NAME

`MAGRIDDeleteByQuery`	–	Deletes 2D or 3D grids from the database meeting the specified criteria.

SYNOPSIS
```
include "MAGRIDAPI.h"
MAIMGRET MAGRIDDeleteByQuery
(
    PMAGRIDDELETEQUERY pQuery,
    long *plNumberDeleted
);
```

INPUT PARAMETERS

PMAGRIDDELETEQUERY pQuery	–	A pointer to a MAGRIDDELETEQUERY structure (section 3.4.24) containing query criteria for the deletions.

OUTPUT PARAMETERS

long *plNumberDeleted	–	A pointer to the number of records deleted.

DESCRIPTION
The MAGRIDDeleteByQuery function deletes from the database all 2D or 3D grid records that satisfy the query criteria provided in the input MAGRIDDELETEQUERY structure. It returns the number of records deleted and the return status structure, MAGRIDRET.

RETURNS
MAGRIDRET Structure – See Section 3.4.18 for details.

NOTES
1. Wildcards are permitted for the following fields in the MAGRIDDELETEQUERY structure: lParameterId, lBeginBaseTime, lEndBaseTime, lBeginTau, lEndTau, lBeginReceiptTime, and lEndReceiptTime.
2. MAGRIDConnect must have been called to start a database session before MAGRIDDeleteByQuery may be called.

SEE ALSO
MAGRIDConnect, MAGRIDRemoteConnect

6.5 MAGRIDUpdateByID

NAME

`MAGRIDUpdateByID` –	Updates the (2D or 3D) grid field for a given datasetname and record id.

SYNOPSIS

```
include "MAGRIDAPI.h"
 MAGRIDRET  MAGRIDUpdateByID
 (
      char *szDataSetName,
      long lRecordID,
      unsigned long ulSizeBytes,
      float *pGridData
);
```

INPUT PARAMETERS

char *szDataSetName	–	The name of the dataset from which the record is to be updated.
long lRecordID	–	An identifier for the record that is to be updated from the named dataset.

OUTPUT PARAMETERS

unsigned long ulSizeBytes	-	The size of the grid field data in bytes.
float *pGridData	-	A pointer to the grid field data points.

DESCRIPTION

The MAGRIDUpdateByID routine takes as input the dataset name, record id, size of grid field blob, and the pointer to the updated grid field. It returns a MAGRIDRET structure indicating the status of the operation.

RETURNS

MAGRIDRET Structure – See Section 3.4.18 for details.

NOTES

1. An update to a MAGRID_BASE grid field type record will automatically generate a new MAGRID_EDITED grid field product record. Base grids may not be modified.
2. Typically a call to MAGRIDGet2DByID, MAGRIDGet2DByQuery, MAGRIDGetVolumeByID must be done first in order to get the grid field to be modified.
3. MAGRIDConnect must have been called to start a database session before MAGRIDUpdateByID may be called.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect, MAGRIDGet2DByID, MAGRIDGet2DByQuery, MAGRIDGetVolumeByID

6.6 MAGRIDVerifyModel

NAME

`MAGRIDVerifyModel`	–	Determines if the registration information exists for a given grid, center, and generating process.

SYNOPSIS

```
#include "MAGRIDAPI.h"
MAGRIDRET MAGRIDVerifyModel
(
      long lProductionCenterId,
      long lSubCenterId,
      long lGridId,
      long lGeneratingProcId
)
```

INPUT PARAMETERS

long lProductionCenterId	-	A Production Center Identifier
long lSubCenterId	-	A Sub Center Identifier
long lGridId	-	A Grid Identifier
long lGeneratingProcId	-	A Generating Process Identifier

OUTPUT PARAMETERS

None.

DESCRIPTION

The MAGRIDVerifyModel routine may be used to determine if a given model's registration exists in the database.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

NOTE

1. The MAGRIDRET nStatus field returns 0 if the registration data exists; otherwise, the value UNKNOWN_REGISTRATION is returned.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect, MAGRIDRegisterModel, MAGRIDRetrRegistration

6.7 MAGRIDRegisterModel

NAME

`MAGRIDRegisterModel` – Registers a grid model into the database.

SYNOPSIS

```
#include “MAGRIDAPI.h”
MAGRIDRET MAGRIDRegisterModel
(
      PMAGRIDREG pRegistrationData
)
```

INPUT PARAMETERS

PMAGRIDINPUTREG pRegistrationData	-	A pointer to a MAGRIDINPUTREG structure (Section 3.4.5) containing Grid Registration data.

OUTPUT PARAMETERS

None.

DESCRIPTION

The MAGRIDRegisterModel routine registers a grid model with the database. The routine updates the center, subcenter, geographic registration, and projection tables with the data contained in the MAGRIDINPUTREG structure. The registration facilitates projection conversions and sub gridding.

RETURNS

MAGRIDRET structure – See Section 3.4.18 for details.

SEE ALSO

MAGRIDConnect, MAGRIDRemoteConnect , MAGRIDVerifyModel, MAGRIDRetrRegistration

7 MAGRID UTILITY METHODS AND FUNCTIONS

7.1 MAGRIDFreeLL

NAME

`MAGRIDFreeLL` – Frees the linked list associated with catalog retrieval and grid retrieval.

SYNOPSIS

```
include "MAGRIDAPI.h"
 int MAGRIDFreeLL
(
    PMAGRIDLINKEDLIST pLL
);
```

INPUT PARAMETERS

PMAGRIDLINKEDLIST pLL	–	A pointer to a MAGRIDLINKEDLIST (Section 3.4.17) linked list of data.

OUTPUT PARAMETERS

PMAGRIDLINKEDLIST pLL	–	Pointer reset to NULL.

DESCRIPTION

MAGRIDFreeLL frees a linked list that has been returned by calls to MAGRIDRetrRegistration, MAGRID2DCatalog, MAGRIDCatRetr, MAGRIDGet2DByQuery, MAGRID3DCatalog, MAGRIDGetTrack, or MAGRIDGridRetr. It returns a status value.

RETURNS

int nReturnValue	–	A status value is returned. A status value of zero indicates a successful completion.

7.2 MAGRIDGetVolumePtr

NAME

`MAGRIDGetVolumePtr` – Gets a pointer into the 3D blob data for the desired level.

SYNOPSIS

```
include MAGRIDAPI.h
 MAGRIDRET MAGRIDGetVolumePtr
 (
      MAGRID3DDATA  st3DData,
      float         rsLevelValue,
      float         **pVolume
);
```

INPUT PARAMETERS

MAGRID3DDATA st3DData	–	The structure that describes the volume of interest.
float rsLevelValue	–	The value of the level for which the pointer is desired.

OUTPUT PARAMETERS

float **pVolume	–	The address of the pointer into the 3D blob.

DESCRIPTION

The MAGRIDGetVolumePtr uses the information supplied in the MAGRID3DDATA structure as well as the provided level value to calculate the starting address of the data associated with that level. It stores this pointer at the given location. It also returns a MAGRIDRET structure indicating the status of the operation.

RETURNS

MAGRIDRET Structure – See Section 3.4.18 for details.

NOTES

1. MAGRIDConnect must have been called to start a database session before MAGRIDGetPoint may be called.
2. MAGRIDGetVolume must have been called prior to a call to MAGRIDGetVolumePtr.

SEE ALSO

MAGRID

7.3 MAGRIDGetPoint

NAME

`MAGRIDGetPoint` – Gets a given point in a 2D Grid Field.

SYNOPSIS

```
include MAGRIDAPI.h
MAGRIDRET MAGRIDGetPoint
(
      float             rsLat
      float             rsLon
      PMAGRIDDATA       pGridData
      PMAGRIDOUTPUTREG  pReg
      float             *prsOutputValue
);
```

INPUT PARAMETERS

float rsLat	– The Latitude that represents the point of interest.
float rsLon	– The Longitude that represents the point of interest.
PMAGRIDDATA pGridData	– The Grid Field Data that were previously retrieved.
PMAGRIDOUTPUTREG pReg	– The Grid registration of the Grid Field Data.

OUTPUT PARAMETERS

float *prsOutputValue	– The pointer to the Grid Value.

DESCRIPTION

MAGRIDGetPoint is a convenience routine that will retrieve a single point of data from a given grid. It takes as input a grid that was retrieved from the database where no registration was applied (MAGRID_GET_AS_STORED). It will apply the registration and pull the point of interest.

RETURNS

MAGRIDRET Structure – See Section 3.4.18 for details.

NOTES

1. Use MAGRIDGet2DByID or MAGRIDGet2DByQuery to fill in the MAGRIDDATA structure that is used as input to MAGRIDGetValue. The output format flag must be set to MAGRID_GET_AS_STORED.
2. Use MAGRIDRetrRegistration to fill in the MAGRIDOUTPUTREG structure that is used as input to MAGRIDGetValue.

(This page intentionally left blank.)

8 NOTES

8.1 Glossary of Acronyms

AESS	Allied Environmental Support System
AOI	Area of Interest
API	Application Program Interface
APIRM	API Reference Manual
COE	Common Operating Environment
DII	Defense Information Infrastructure
GCCS	Global Command and Control System
IC4ISR	Integrated Command, Control, Communications, Computer, and Intelligence Surveillance Reconnaissance
IMOSS	Interim Mobil Oceanographic Support System
JMCIS	Joint Maritime Command Information System
JMS	Joint METOC Segment
LLT	Latitude-Longitude-Time
MAGRID	Grid Field API Segment of the TESS(NC) METOC Database
MDGRID	Grid Field Database Segment of the TESS(NC) METOC Database

METOC	Meteorological and Oceanographic
MIDDS	Meteorological Integrated Data Display System
NITES	Navy Integrated Tactical Environmental Subsystem
PC	Personal Computer
PM	Programming Manual
PS	Performance Specification
SPAWAR	Space and Naval Warfare Systems Command
SQL	Structured Query Language
TEDS	Tactical Environmental Data System
TESS(NC)	Tactical Environmental Support System Next Century